# Choreography-Driven Socialization of Peer-to-Peer Communities

Xi Bai, XiaLi Li, Dave Robertson

Centre For Intelligent Systems and their Applications, School of Informatics, University of Edinburgh

*xi.bai@ed.ac.uk*

## Abstract

From the perspective of choreography, an approach for forming and evolving Peer-to-Peer (P2P) communities is proposed in terms of interactions between peers. Since community can prune the peer search space, relying on it, a peer can more easily discover the desired services and other collaborative peers. An Interaction Model (IM) publication strategy is presented, based on which users can publish their IMs without exposing their private and sensitive knowledge to others. Except ontologies for describing the structure of P2P communities, there is no other unified ontology required in our approach, so peers can use arbitrary ontologies and corresponding matchmakers to make use of heterogeneous resources. A simulation of service discovery and community formation is presented as well, indicating that our approach is generic and will facilitate the emergence of various peer side applications.

## Choreography Description Using Interaction Model

From the perspective of choreography, we use Lightweight Coordination Calculus (LCC) to describe the interactions between pees. Following LCC codes depicts an Interaction Model (IM) in which a client purchases a product from a shop using his or her credit card but since there is no explanation about any element belonging to this IM, it is impossible for peers that want to purchase or sell a product to automatically recognize this IM which is exactly the one they really need. Therefore, service discovery and reuse are both hampered.

$r(client, initial, 1)$
$r(shop, necessary, 1)$

$a(client(ProductCode), C) ::$
$\quad buy(ProductCode, CreditCard) \Rightarrow a(shop, S) \leftarrow cc(CreditCard) \wedge$
$\qquad\qquad lookup(S) then$
$\quad receipt(Receipt) \Leftarrow a(shop, S)$

$a(shop, S) ::$
$\quad buy(ProductCode, CreditCard) \Leftarrow a(client(\_), C) then$
$\quad receipt(Receipt) \Rightarrow a(client(\_), C) \leftarrow$
$\qquad enough\_credit(CreditCard, ProductCode) \wedge$
$\qquad complete\_order(ProductCode, CreditCard, Receipt)$

## P2P Community Ontology

We give a brief description about our P2P community ontology. In this ontology, two notable points are the notion of *Peer* and the notion of *Community*. The behaviors of a peer are just like the ones a person has in the real world, e.g., a peer may knows about other peers in the network. Since *FOAF* vocabulary is a widespread ontology used for describing persons, their activities and their relations to others, we take advantage of this vocabulary to describe the peers as well. A community is a group of peers which have some common interests. We want peers belonging to the community can do interconnecting discussions to each other through some methods such as blogs, forms and mailing lists in the future. For the sake of this purpose, we maximize the extensibility of our community notion by using *SIOC* vocabulary. Base on the above analysis, *openk:Peer* is the sub-class of *foaf:Person* and *openk:P2PCommunity* is the sub-class of *sioc:Usergroup*. Peer profiles are very important for community formation and our *Peer* class is like a template that defines the structure of the peer profile. The IM ontologies were extracted in terms of the natural structure of LCC codes, which are very intimate with the above P2P community ontologies. The concise structure of our P2P Community Ontology is described in Figure 1.

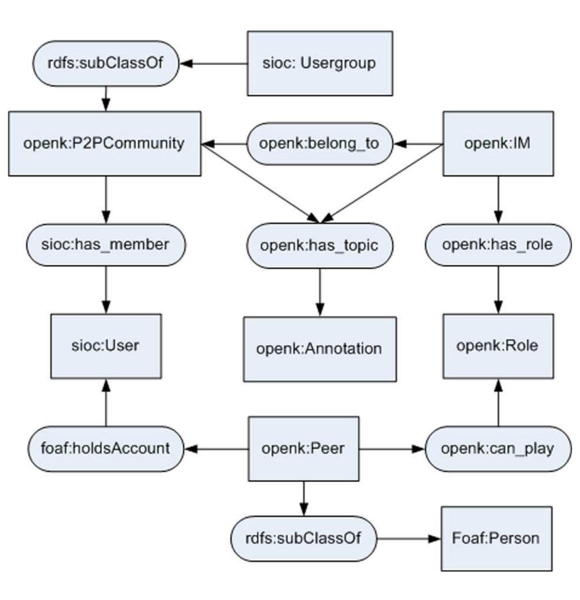


Figure 1: P2P Community Ontologies

Then the IM described in the last section can be annotated in terms of P2P community ontologies using RDFa or Microformats.

## Community Formation

A *Community* is a group of peers which share common interests and are capable of collaborating with each other to fulfill one or more specific IMs related to this community. The IM for forming P2P communities is described as follows:

$role(group, initial, 1)$
$role(subscriber, necessary, 1)$
$role(member, necessary, 1)$
$a(group(GPS, GPF), G) ::$

$\quad (enter(PPS) \Leftarrow a(subscriber, Ps) then$
$\quad a(group(GPS, GPF), G) \leftarrow not(teminate(G, GPS)) \wedge$
$\qquad\qquad recruit(Ps, G, GPS, PPS, GPF))$
$\quad or$
$\quad (exit(PPS) \Leftarrow a(member, Pm) then$
$\quad a(group(GPS, GPF), G) \leftarrow remove(Pm, G, GPS, PPS, GPF))$
$\quad or$
$\quad (merge \Leftarrow a(group(EGP, \_), E) then$
$\quad a(group(GPF, GPF), G) \leftarrow merge(G, E, GPS, EGP, GPF))$
$\quad or$
$\quad (exec(AS, IM, PL) \leftarrow member\_of(G, IM, GPS) \wedge$
$\qquad\qquad quorate(IM, PL, GPS) \wedge initiator(IM, PL, AS, GPS) then$
$\quad (group(GPS, GPF), G))$
$\quad or$
$\quad (merge \Rightarrow a(group(EGP, \_), Y) \leftarrow merge\_condition(GPS, Y) \wedge$
$\qquad\qquad GPF = GPS)$
$\quad or$
$\quad (null \leftarrow terminate(G, GPS) \wedge GPF = GPS).$

$a(subscriber, Ps) ::$
$\quad enter(PP) \Rightarrow a(group(\_, \_), G) \leftarrow want\_to\_subscribe(Ps, G).$

$a(member, Pm) ::$
$\quad exit(PP) \Rightarrow a(group(\_, \_), G) \leftarrow want\_to\_leave(Pm, G).$

## Community Evolution

The IM for P2P community evolution is described as follows:

$role(peer, initial, 1)$
$role(helper, necessary, 1)$

$a(peer(PPS, PPF), Pi) ::$
$\quad update(ERs) \Leftarrow r(helper(ERs), H) then$
$\quad a(peer(PPS, [\,], LRs, PPF, ERs, ), Pi) then$
$\quad update(LRs) \Rightarrow a(Peer(\_, \_), Pe) \leftarrow want\_to\_share(LRs))$

$a(peer(PPS, U, LRs, PPF, ERs), Pi) ::$
$\quad (null \leftarrow ERs = [\,] \wedge PPF = PPS)$
$\quad or$
$\quad (null \leftarrow ERs = [Rule|ERs_R] \wedge inference(Rule, PPS, U)$
$\quad \wedge merge\_rule(Rule, PPS, LRs) then$
$\quad a(peer(U, [\,], LRs, PPF, ERs_R), Pi))$

Based on the aforementioned P2P community ontologies, we can provide peers especially who have expertise with a chance to create rules that will direct themselves and other peers to update their local profiles.

*I.* $\forall P \forall IM.publish\_IM(P, IM)$
$\rightarrow \exists R.can\_play(P, R) \wedge has\_role(IM, R)$

*II.* $\forall P \forall IM \forall C.publish\_IM(P, IM) \wedge belong\_to(IM, C)$
$\rightarrow \exists U.holdsAccount(P, U) \wedge has\_member(C, U)$

*III.* $\forall P \forall R \forall IM \forall C.can\_play(P, R) \wedge has\_role(IM, R) \wedge belong\_to(IM, C)$
$\rightarrow \exists U.holdsAccount(P, U) \wedge has\_member(C, U)$

*IV.* $\forall P \forall R.(\forall T.has\_constraint(R, T)$
$\rightarrow can\_satisfy(P, T)) \rightarrow can\_play(P, R)$

*V.* $\forall IM \forall .C(\forall A.has\_annot(C, A)$
$\rightarrow has\_annot(IM, A)) \rightarrow belong\_to(IM, C)$

*VI.* $\forall IM \forall R \forall A.has\_role(IM, R) \wedge has\_annot(R, A)$
$\rightarrow has\_topic(IM, A)$

*VII.* $\forall IM \forall A \forall C.has\_topic(IM, A) \wedge belong\_to(IM, C)$
$\rightarrow has\_topic(C, A)$

*VIII.* $\forall P_A \forall R_A \forall P_B \forall R_B \forall IM.can\_play(P_A, R_A) \wedge has\_role(IM, R_A) \wedge$
$can\_play(P_B, R_B) \wedge has\_role(IM, R_B) \wedge P_A \neq P_B \wedge R_A \neq R_B$
$\rightarrow knows(P_A, P_B)$

## Simulation

In a P2P environment, the basic and smallest interaction happens between two peers. Therefore, we can use PCs to simulate the invitation process between peers. Each computer has been installed with an server that interacts with other servers running on other peers by sending out or receiving messages. This makes each computer a client and server at the same time. IMs belonging to each peer are published at each server side. The topology of the P2P community is depicted in Figure 2.

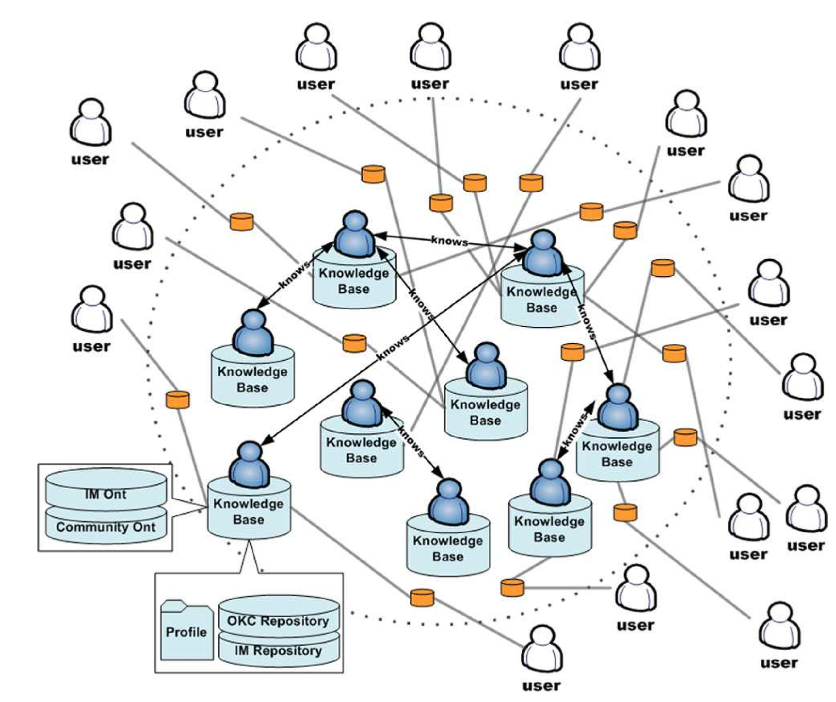


Figure 2: P2P Community Topology

Suppose *Peer A* expects to invite *Peer B* to join its community by sending out an IM. Then *Peer B* receives this IM by browsing the corresponding advertisement page running on the server of *Peer A*. An application at the side of *Peer B* can automatically parse the IM page into RDF statements. By querying these RDF statements, *Peer B* can check if it can play roles in the IM. If the answer is true, *Peer B* will further query the KB stored on *Peer A* to find out if this IM belongs to some specific communities or not and apply for a membership if possible. The URI of this IM is denoted by $im\_uri$; the named graph of KB on *Peer B* is denoted by $PeerB\_KB$; the URI of the graph of received IM is denoted by $IM$. Then the following Open Knowledge Component (OKC) implemented using SPARQL can help *Peer B* figure out which role it can play and which potential community it can join:

```
SELECT ?role ?community
FROM ⟨IM⟩
FROM NAMED ⟨PeerB_KB⟩
WHERE {
  GRAPH ⟨PeerB_KB⟩ {
    ?PeerB a openk:Peer.
    ?role a openk:Role.
    ?PeerB openk:can_play ?role.
  }
  ?community a openk:P2PCommunity.
  ⟨im_uri⟩ openk:has_role ?role.
  ⟨im_uri⟩ openk:belong_to ?community. }
```

A peer being capable of playing a role may or may not be declared explicitly in its profile, so rules or composite rules can be used for mining this implicity. According to Rule *IV*, "?PeerB openk:can_play ?role" in the above OKC can be inferred by invoking the following OKC:

```
result_set = SELECT ?peer ?role
WHERE {}
for each result in result_set
  flag = ASK WHERE {
      ⟨result.getURI(?peer)⟩ openk:can_play
      ⟨result.getURI(?role)⟩}
  if flag == false
    result_set′ = SELECT ?constraint
    WHERE {
      ⟨result.getURI(?role)⟩ openk:has_constraint
      ?constraint
    }
    CONSTRUCT {
      ⟨result.getURI(?peer)⟩ openk:can_play
      ⟨result.getURI(?role)⟩}
    WHERE {
      for each result′ in result_set′
        ⟨result.getURI(?peer)⟩ openk:can_satisfy
        ⟨result′.getURI(?constraint)⟩.
      end-for
    }
  end-if
end-for
```

## Future Work

Currently IM publishers are allowed use RDFa to manually inject annotations into XHTML. Based on the annotation embedded page, we can develop various peer side applications such as IM discovery and community member recruitment. A middle ware should be provided to publishers for publishing and annotating their IMs semiautomatically. The P2P community formation approach needs to be evaluated in the future by simulating a large scale P2P network.